Technical Report

The Roman Conquered by Delay: Reducing the Number of Preemptions using Sleep States

Muhammad Ali Awan

Stefan M. Petters

HURRAY-TR-110301

Version:

Date: 03-15-2011

The Roman Conquered by Delay: Reducing the Number of Preemptions using Sleep States

Muhammad Ali Awan

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

http://www.hurray.isep.ipp.pt

Abstract

Sleep-states are emerging as a first-class design choicein energy minimization. A side effect of this is that the releasebehavior of the system is affected and subsequently thepreemption relations between tasks. In a first step we haveinvestigated how the behavior in terms of number of preemptions of tasks in the system is changed at runtime, using an existing procrastination approach, which utilizes sleepstates for energy savings purposes. Our solution resulted in substantial savings of preemptions and we expect from even higher yields for alternative energy saving algorithms. This work is intended to form the base of future research, which aims to bound the number of preemptions at analysistime and subsequently how this may be employed in the analysis to reduced the amount of system utilization, which is reserved to account for the preemption delay.

The Roman Conquered by Delay: Reducing the Number of Preemptions using Sleep States*

Muhammad Ali Awan Stefan M. Petters
CISTER Research Unit
ISEP-IPP
Porto, Portugal
{maan,smp}@isep.ipp.pt

Abstract

Sleep-states are emerging as a first-class design choice in energy minimization. A side effect of this is that the release behavior of the system is affected and subsequently the preemption relations between tasks. In a first step we have investigated how the behavior in terms of number of preemptions of tasks in the system is changed at runtime, using an existing procrastination approach, which utilizes sleep-states for energy savings purposes. Our solution resulted in substantial savings of preemptions and we expect from even higher yields for alternative energy saving algorithms. This work is intended to form the base of future research, which aims to bound the number of preemptions at analysis time and subsequently how this may be employed in the analysis to reduced the amount of system utilization, which is reserved to account for the preemption delay.

1 Introduction

Embedded systems have become a necessity of our normal life. Typical examples of such system are GPS, music systems or mobile phones. Real-time (RT) embedded systems have additional timing constraints. Not only the functionality but also timing requirement should be met for the correctness of the system. In some special cases deviation from the timing constraint could be disastrous. For example safety critical applications in cars i.e. airbag.

Apart from functionality and/or timing constraints, many embedded system have a limited energy supply such as battery powered mobile devices or devices with limited or intermittent power supply, e.g. solar cells. The hardware designers have provided several features for the system designer to explore energy savings, such as dynamic voltage and frequency scaling (DVFS) and sleep states.

The recent technology trends have equipped the modern embedded processors with the several sleep states and reduced their overhead (energy/time) of the sleep transition. The DVFS potential to save energy is diminishing due to efficient (low overhead) sleep states and increased static power consumption. Le Sueur and Heiser [1] have found that race-to-halt followed by a sleep state is emerging as a candidate superior to DVFS for energy management purposes. In a race-to-halt solution tasks are executed as fast as possible and the CPU is sent to a sleep state on conclusion of a job of the task. The sleep state is usually initiated for a predefined interval, as the overhead of transitioning into and out of a sleep state are still substantial.

The forced sleep-state interval gives rise to two conflicting constraints. 1) On one side, execution of the tasks releases during the sleep-state interval are delayed and constrained to a smaller window for execution. One could easily perceive that number of preemptions will rise, as delaying the tasks execution increase the likelihood of higher priority tasks releases. Thus in the presence of low priority tasks, higher priority tasks cause more preemptions. 2) On the other side, the interrupts that occur throughout the sleep state interval are served on completion of the sleep interval. We consider that a task release is triggered by an interrupt. Therefore, tasks releases during sleep interval are bunched together and scheduled after the sleep state. Thus delaying new tasks arrival and waiting for the higher priority task releases decreases the number of preemptions. Thus these two considerations indicate positive or negative changes in the number of preemptions.

For the scope of this paper, we assume the preemption count, represents the count of preemptions taking place when a actively executing task is being replaced before it has completed execution by a higher priority task to execute. Suppose we have synchronous releases of high and low priority tasks. The high priority task executes and preemption is not counted, as low priority task has not yet

*This work was supported by the Portuguese Science and Technology Foundation (FCT) (RePoMuC project PTDC/EIA-EIA/112599/2009) and the ARTEMIS-JU (RECOMP project ARTEMIS/0202/2009).

started its execution.

The number of preemptions poses a substantial overhead on the running system. On resumption of a task the system has to pay the penalty to reload the cache content displaced by preemption. Access to off-chip memory is generally very expensive when compared to on-chip caches or scratch-pad memory. The decrease in the number of preemptions is resulting in a reduction of overall system utilization and subsequently of energy consumption.

Considering the overhead of preemption on the energy consumption, it is indeed an important issue to resolve which constraint overwhelm. This issue raises many questions. 1) The forced sleep increases or decreases the number of preemptions? 2) If the number of preemptions is increased, the overall energy saved from the sleep transition is more or less than the energy penalty caused by the extra preemptions? Nevertheless, if the number of preemptions decreases, the overall energy consumption actually decreases more than just energy saved though sleep transition, as we reduced the overhead of preemptions.

2 System Model

We assume implicit-deadline sporadic task model, with ℓ independent tasks. The task set is $\mathrm{T} = \langle \tau_1, \tau_2, \ldots, \tau_l \rangle$. A Rate-Based Earliest Deadline first (RBED) framework [2] is used to schedule the jobs released by T. RBED is based on the earliest deadline first (EDF) scheduling algorithm. A task τ_i is represented by a triplet $\langle C_i, T_i, A_i \rangle$, where C_i is the worst-case execution time (WCET), A_i is the periodic budget allocated to the task and T_i is the inter-arrival time and relative deadline of each job of the task. Our independent tasks are released as a sequence of jobs. We assume a varying execution time of jobs.

The misbehaviour under overloaded conditions is one of major limitations of EDF scheduler. The RBED framework [2] is used to provide a temporal isolation via enforced budgets A_i associated with each job. This temporal isolation allows the mixing of hard real-time, soft real-time and best-effort type applications. For hard real-time (HRT) tasks budget is equal to the WCET ($A_i = C_i$), to ensure the timely completion of all jobs. The allocations of budget for best-effort (BE) tasks can be independent of their WCET i.e. $A_i \leq C_i$. Normally, BE tasks execute for less than their allocated budget A_i and the remaining budget is considered as execution slack. However, BE task may require more than its allocated budget, which will lead to an overloaded situation. The RBED takes precautionary measures to deal with such condition, opposed to classical EDF which may have drastic consequences. The scheduler preempts every job when it has used up its allocated budget a_i. The deadline of the task is extended for one period and its budget is refilled again. The remaining portion of its execution is postponed until it is scheduled again. Thus a BE job exceeding its budget cannot affect the overall schedulability. The interested reader is directed to the original work of Brandt et al. [2,3] for a detailed discussion and corresponding proofs.

The different needs and varying slack situations in embedded systems motivated the hardware vendors to provide the platforms with multiple sleep states. To model our system closer to reality, we assume N sleep states in the system. Each of these is associated with a transition overhead in time and energy. Static slack in our model is the spare capacity in the schedule to a system utilization U_i being less than 1. Execution slack is generated from a difference of C_i and the actual execution time of a job. The sporadic slack corresponds to the delay due to sporadic release of tasks.

3 Experimental Setup

We used a discrete event simulator to implement the Leakage Control Earliest Deadline First (LC-EDF) algorithm [4]. LC-EDF is based on procrastination scheduling and initiates sleep, when the system is idle. It relies on external hardware to recompute the sleep interval when tasks arrive while the processor is in a sleep state. The interested reader is directed to the original publication [4] for more details. The only modification we made in the algorithm, is the selection of sleep state based on task-set utilization. In the original work, the system has three states: executing, idle or sleep mode. However, as we assume N sleep states, we select for each run the appropriate sleep state among the available set of sleep states based on the maximum feasible idle interval. The system utilization is varied from 0.1 to 1 with an increment of 0.02. The task set size is chosen to be $|\mathrm{T}| \in \{20, 30, 40, 50\}$. The T_i of HRT and BE task is specified within a range of $[30ms, 50ms]$, $[50ms, 1sec]$ respectively. The share of HRT and BE tasks in task-set size and system utilization is $\langle 60\%, 40\% \rangle$.

Beyond those initial settings a two level approach is used for generating a wide variety of different tasks and subsequently varying jobs. Tasks are further annotated with a limit on the sporadic delay Δ_i^s in the interval $[0, \Gamma * T_i]$ and on the best-case execution time C_i^b in the interval $[\xi * C_i, C_i]$. Where, $\xi \in \{0.25, 0.5, 0.75, 1\}$ and $\Gamma \in \{0, 0.2, 0.4, 0.6\}$. However, not only tasks vary in their requirements, the same task is also varying behavior dependent on system state and input parameters. This is modeled, by assigning each job an actual sporadic delay in the interval $[0, \Delta_i^s]$ and an actual execution time in the interval $[C_i^b, C_i]$. Considering all above mentioned parameters, 552 task sets are generated. For each task set, the seed value of the random number generator is varied from 1 to 100. In total, we simulated 55200 combinations of all different parameters. Each of them is simulated for 100 sec. The results for every task set are averaged over all seed values to get single value.

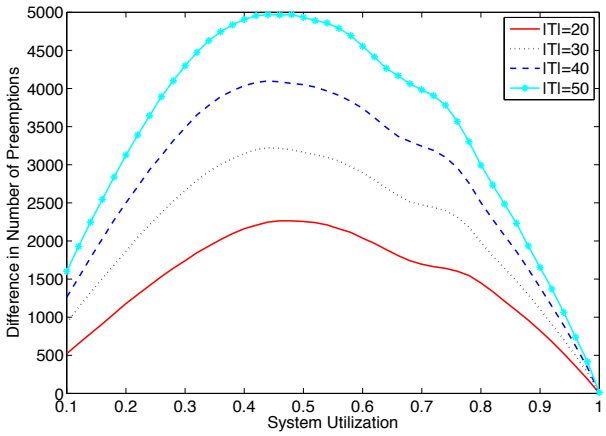

Figure 1. Difference between the absolute number of preemptions of EDF and LC-EDF and $\xi = 0.5$

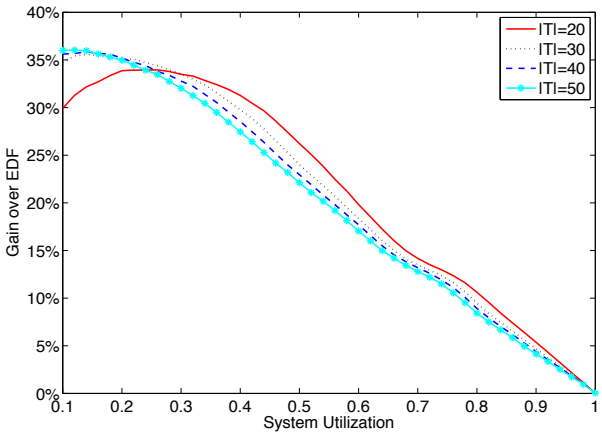

Figure 2. Percentage gain of LC-EDF over EDF with respect to number of preemptions and $\xi = 0.5$

4 Results

The LC-EDF procrastinates the tasks execution when the task arrival occurs while the processor is in a sleep state. This behavior is in-line with our requirement to observe the effect of tasks procrastination over the number of preemptions. The LC-EDF concentrates the future tasks arrival and on the other side pushes the low priority tasks closer to the future releases of higher priority tasks. The number of preemptions are counted for EDF and LC-EDF based on the parameters defined in the system model. We illustrated the results in absolute numbers of preemptions as well as in percentages gain of LC-EDF over EDF with respect to number of preemptions.

The simulations shows that procrastination of tasks execution, and bunching them together actually reduces the number of preemtions in dynamic priority systems. This can be explained through the following reasoning. While logically pushing the release of several tasks into a single instant, the tasks collected do not preempt each other in the classical sense, as none of the tasks arrived has started execution before the arrival of higher priority tasks, which enter the scheduler ready queue in the same instant. Outside the sleep interval, the preemption relations are not affected, leading to an overall reduction of preemption count.

The absolute difference between the number of preemptions of EDF and LC-EDF is represented in Figure 1 for four different task-set sizes. At low utilizations, the number of preemptions is lower in both cases due to larger static slack in the system. As the utilization increases the difference increases, because LC-EDF can save preemptions by not allowing execution for predefined interval. With a further increase in utilization, the sleep interval enforced by LC-EDF reduces as well. Thus it cannot delay the execution for longer interval and therefore difference decreases. At utilization of 1, LC-EDF cannot afford any forced sleep state, even if there would be dynamic slack available in the system. Therefore it behaves similar to EDF. As can be observed in the results, there are slight deviations to an otherwise smooth curve dependent on the system utilization. As discussed in the experimental setup, we have used a number of sleep states and higher system utilization leads to certain deep sleep states to be infeasible.

Another prominent effect in Figure 1 is the difference between different task-set sizes. The increase in task-set size, raises the number of preemptions in EDF, and consequently the potential for LC-EDF to combine more tasks. Therefore difference between these two approaches increase, with large task sets. Figure 2 further illustrates the percentual gain of LC-EDF over EDF when considering the number of preemptions for four different task-sets sizes. It demonstrates that an increase in task-set size, actually reduces the percentual gain over EDF after a utilization of 0.25. As the number of total preemptions in EDF is increasing at a higher rate than the difference of preemptions between EDF and LC-EDF. Another point worth mentioning here is the fact that, the gain over EDF decreases with an increase in utilization. This is due to a reduce potential of LC-EDF to force sleep state for longer interval.

We also observed the effect of varying execution time and sporadic delay of the tasks on the number of preemptions. By increasing the value of ξ and consequently the C_i^b, we are increasing the effective system utilization by reducing the interval between C_i^b and C_i. Thus an increase in ξ will decrease the number of idle intervals (as execution slack decreases), and subsequently a decrease in number of forced sleep intervals. The data in Figure 3 reflects this by a drop in the percentual gain with an increase in ξ. Similar trends are observed in Figure 4 with the variation of sporadic delay. In general extra slack whether caused by

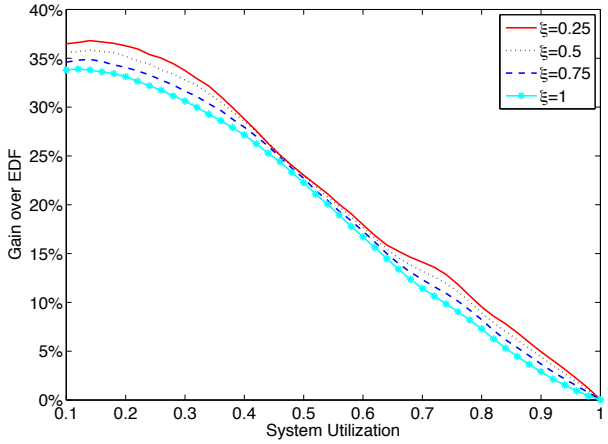

Figure 3. Percentage gain of LC-EDF over EDF with varying execution time delay limit ξ and $|T| = 40$

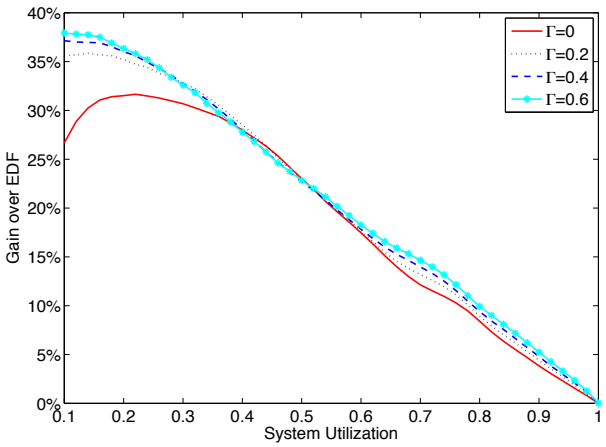

Figure 4. Percentage gain of LC-EDF over EDF with varying sporadic delay limit Γ and $|T| = 40$

increased sporadic delay or lower best-case execution time allows more sleep transitions, and hence the potential for reduction in number of preemptions.

5 Future Work

The recent work of Bertogna and Baruah [5] addressed the preemption reduction and proposed limited preemption EDF scheduling for a sporadic task model. The static slack available in the system is used to avoid unnecessary preemptions, while guaranteeing schedulability. Opposed to their approach, this work does not aim at delaying preemptions as a general measure as proposed by Bertogna and Baruah. Our work will rather exploit the slack in the system for energy management purposes and hence, it may not reduce the number of preemptions to the same degree. However, as the reduction of preemptions comes as a side effect, it is indeed a welcome improvement.

As a future consideration, we are interested to find the bounds on the number of preemptions saved while delaying the schedule with sleep intervals. The main target is to analytically relate a sleep interval with the number of preemptions. Moreover we will investigate other sleep-state based energy management approaches with explicit slack management, and whether those provide favourable guaranteeable reductions in the number of preemptions. Finally, we will also explore how these reduced preemptions can be exploited in the analysis and run-time of the system.

6 Conclusions

In this paper, we observed the effect of sleep state on the number of preemption for dynamic priority systems. We found that delaying the task execution of low priority tasks and waiting for the higher priority tasks that can potentially preempt the previous low priority tasks, reduces the number of preemptions. This observation will allow the system designer to further reduce the overall system energy consumption, as preemptions impose an extra overhead on the energy consumption. However, there is still need to develop more efficient power saving approaches based on sleep states, as LC-EDF itself is based on a very unrealistic assumption of an external hardware to implement their algorithm. The increase in leakage power consumption, efficient sleep state in modern processor and their effect shown on number of preemptions, reflects the importance of sleep states over DVFS. Furthermore, DVFS increases the tasks execution time and in turn probability of more preemptions.

References

[1] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proceedings of the 2010 Workshop on Power Aware Computing and Systems*, (Vancouver, Canada), Oct 2010.

[2] S. A. Brandt, S. Banachowski, C. Lin, and T. Bisson, "Dynamic integrated scheduling of hard real-time, soft real-time and non-real-time processes," in *Proceedings of the 24th IEEE Real-Time Systems Symposium*, (Cancun, Mexico), Dec. 2003.

[3] C. Lin and S. A. Brandt, "Improving soft real-time performance through better slack management," in *Proceedings of the 26th IEEE Real-Time Systems Symposium*, (Miami, FL, USA), Dec. 2005.

[4] Y.-H. Lee, K. Reddy, and C. Krishna, "Scheduling techniques for reducing leakage power in hard real-time systems," in *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, pp. 105 – 112, jul. 2003.

[5] M. Bertogna and S. Baruah, "Limited preemption edf scheduling of sporadic task systems," *Industrial Informatics, IEEE Transactions on*, vol. 6, pp. 579 –591, Nov. 2010.